Global Journal of human-social science: G
Linguistics & Education
Volume 20 Issue 4 Version 1.0 Year 2020
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 2249-460X & Print ISSN: 0975-587X

Implementation of a Verbal Compiler: The Need to Develop Audio Language to Keep Pace with Rapid Development becomes a Necessity

By Laarfi, Ahmed & Dr. Kepuska, Veton

Abstract- This research paper aims to make essential developments in Speech Recognition(SR), the compiler gives the user a choice to choose the type of output, whether it is textual or conversational (audio). Many large companies have developed such Speech Recognition Systems (SRS), especially the companies producing Smartphones, Computers, and Laptops. If translation is taken as a model application, they have not yet developed the perfect systems. The purpose of this paper is to add facilities to the Speech Recognition (SR) software so that it can deal with spoken languages.

Index Terms: artificial intelligence, speech recognition, compiler construction, audio programming language.

GJHSS-G Classification: FOR Code: 200499

Strictly as per the compliance and regulations of:

Implementation of a Verbal Compiler: The Need to Develop Audio Language to Keep Pace with Rapid Development becomes a Necessity

Laarfi, Ahmed [α] & Dr. Kepuska, Veton [σ]

"Great achievements begin with small dreams"

Abstract- This research paper aims to make essential developments in Speech Recognition(SR), the compiler gives the user a choice to choose the type of output, whether it is textual or conversational (audio). Many large companies have developed such Speech Recognition Systems (SRS), especially the companies producing Smartphones, Computers, and Laptops. If translation is taken as a model application, they have not yet developed the perfect systems. The purpose of this paper is to add facilities to the Speech Recognition (SR) software so that it can deal with spoken languages.

Index Terms: artificial intelligence, speech recognition, compiler construction, audio programming language.

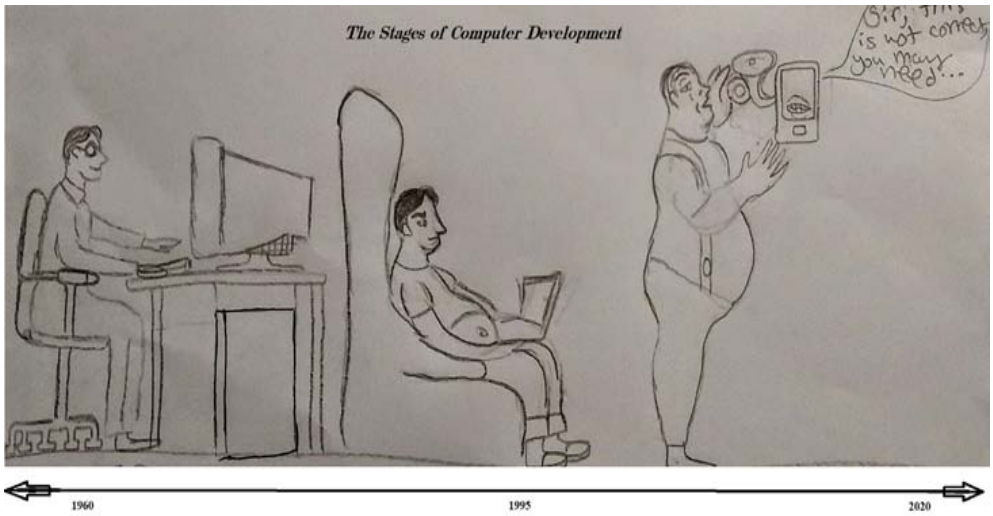


Fig.1: From Electronic Numerical Integrator and Computer (ENIAC), a building, to where? *

I. Preface

a) *"Two Chineese speak chinheese, but they need a chinheese to chinheese interpreter process"*

SR is defined as speaking to computers in any language. The process is described through complex technical operations, which will be addressed later. The number of software packages that utilize SR has been massively increased. Accordingly, the software are commonly used. Their importance is increasing because many devices use this new paradigm. The software have become essential applications of Laptops and Smartphones and other critical electronic devices. The Compiler is a virtual part like an interpreter between two people who speak entirely different languages. Most of the traditional translators in programming languages are textual. Some changes have been made to enhance the translators by entering other types of files such as sounds, graphics, and videos in different forms and extensions. Known computer languages that are numbered in the tens, if not hundreds, accurately represent the compiler. All compilers rely on the rules of a language, which are fixed but mostly similar to each other. The compilation of the artificial intelligence of SR is quite like a person who speaks Japanese in a local dialect that is not known even to the other Japanese. Many translators who have studied Japanese grammar are often unable to translate it correctly. The intended use of such applications to advance the most important modern technologies is advocated.

Author α: A Ph.D. Candidate of Computer Engineering, Artificial Intelligence. e-mail: Alaarfi2015@my.fit.edu

Author σ: A Professor of Artificial Intelligence, A speech Recognition, School of Electrical Engineering and Science, FIT, 150 W. University Blvd, Melbourne, FL, 3290. e-mail: vkepuska@fit.edu

Fig. 2: Bridging The Different Language*

II. Introduction

a) *"We live the future in our present"*

One of the most important features provided by the audio input of the computer is that it frees users from several activities. For example, there is no longer a need to use hands or any other type of input. Moreover, a quick insertion-input method of input frees the eyes from focusing on the input mode.

Of course, such applications have many problems since their launch four decades ago until today. The acceleration of technological developments leads to improved quality of such applications. Users rely more and more on usage of such applications that utilize SR. The spelling of commands and the commonly used input units are utilized. A separate word may be stored in a variable of a specific type declared by the compiler designer. Symbols, erratic phrases, or related sentences can be stored as variables according to the types of the entered data. The storage is completed by interaction between the compiler and the human. Ultimately, the final confirmation should be affirmed by the yes or no question. Thus, we have variables stored in memory addresses. After storage, a kind of artificial intelligence is introduced that leads to continual updating whether correcting previous errors or keeping the data available while storing any new information. For example, if a date is requested, 8 digits expected to be entered (2 digits for the day, 2 digits for the month, and 4 digits for the year). In the 1990s, the computer whose memory was 2 megabytes was considered a good machine, but 4 megabytes was excellent. At that time, video files were the most resource-intensive, followed by high-resolution image files. Audio files could not be ignored in terms of required allocated space. The text files are the smallest in terms of memory size. Nowadays, computers have vast space available without considering the additions of hard disks and external storage devices. Although abundant memory space is available, the concern is the files' sizes being as small as possible. Therefore, the voice information is stored as text data; this feature provides users with the ability to handle input/output files as audio files at the same time, that is, as byte/text files.

PART 1 What to Do.

b) *"Programming languages produce other programming languages."*

A. Design and implementation of a compiler includes a language supported by speech recognition in general

In the late 1970s, a significant development took place in speech recognition systems (SRS), a vast improvement but still not enough to build a complete system that deals with SR. The improvements in this field have been continuously occurring whereby the use of software instead of hardware programming is utilized. In the second decade of the 21st century, SR is used everywhere. The SRs are easily programmed in C++ language in the Visual Studio group by Click & Drag through a box that acts as a voice recorder to save the data in any audio file format. Converting the text file to a voice file in the type of format needed is simple. In some Windows versions, a facility of recording voices by using the speaker icon is found on the toolbar. Also, can easily read text files as audio. Generally, most applications, whether audio or otherwise, have characteristics of the conversion of files from a format to another format that can utilize applications like Excel, pdf, and Word. Furthermore, software packages that perform a specific task, such as Laser and Photonics, can convert simulated files to data files in the form of images and vice versa to be used later as input either in the same or another software. An example is converting file extensions between Optiwave and Optic Studio.

As of today, the word processor accepts input in audio form in many languages. Numerous software packages prepare, write, pronounce, or convert what was read to a text file, making communication between people who cannot interact linguistically together much more relaxed. We need to translate from one language to another. Such software is still largely incapable of translation even with languages of the same family. We are also faced with the problem of ambiguity.

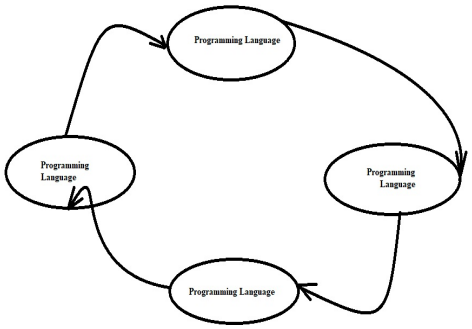


Fig. 3: Recursive means that the compiler is any high-level language that can program a set of programs. By the new compiler, another new compiler can be produced

B. Why and how do we design Compilers in the middle of the great congestion between programming language interpreters & compilers?

c) *"Using these systems, our breaths become counted"*

I have not heard or used a programming language in which the commands are uttered. Software to search for something or somewhere receives orders by voice. Automated voice systems control many activities, including schedules or cancelations and voice payment systemsthat give access to the software that monitors the calls by recording them. Numerous other reasons include quality assurance. SR usage represent examples of systems programmed by a specific language.

Compilers are similar to any other executive program that ends with the extension .com or .exe gives programmers the environment to design and provide all programming needs based on the purpose of the program. Compilers orders and divides the files into packages (units) and performance-related tasks. Inside each are classes that contain procedures, functions (methods), and all types of compiler stages (lexical, syntax, and semantic), and in order, may express implementation types like defining variables, records, files, arrays, libraries, and all kinds of saved files. In other words, the compiler is also a program that uses a high-level source language to transform into a low-level target language. While running on its environment, the source compiler should detect errors, report them, and inform the programmer to make required corrections.

Second: The lack of SR applications in a language such as Arabic leads to the need to find solutions and alternatives, either by producing a spoken programming language or by developing the few available applications. In this paper, all emphasis is placed on the creation of a spoken language by designing a compiler that achieves this purpose.

The main program, which stays resident in the memory, should have a minimum size, and be limited to few commands that apply the Dynamic Loading Technique, which calls the target program, bring it from its physical address to store temporarily in the main memory, performs its task, and returns to its location outside the memory. The commands that use this operating system technique should be available and implemented in the language used. Once the compiler runs, it shows, edit, save and modify commands on the main screen. A user-friendly screen with menus helps programmers to perform tasks. Visual languages are preferable.

As discussed before, the SR files are used as input/output files or voice/text files, and the conversion between them. Most languages serve the call of such files either way and even the text, if it is not lengthy, can be edited in the command itself if programmers decide to write manuscripts to be read as voice files.

Fig. 4: The Amphibious language*

PART II: A Frog, Amphibious, Language

d) *"All that we are seeking is to command the computer system acoustically to obey"*

A New Generation of Language Programming: a programming language that receives voice commands based on the mechanisms of SR.

System Definition Without detailing the nature of sound production, technical details of the relationship between humans and computers in terms of SR are discussed. Mathematical calculations are vital to solving problems. The figure provided below demonstrates the process of how speech should be utilized. Designing a compiler to deal directly with the SRSs is vital, particularly due to the advancement of the Modern Generation of Programming Languages (MGPL) that support voice commands as an alternative to writing. Besides, such a system develops and simplifies programming languages so that words can be used from the spoken language of programmers. Moreover, SR increases the speed of achievement, avoiding the incompatibility between the Speech Recognition Techniques (SRT), and the compilers. In many common components, exchanging procedures may be neededfor SR and Compiler Construction (CC). Instead of adapting the two systems to work in one environment, the common framework would be appropriate for work together from the scratching with complete compatibility.

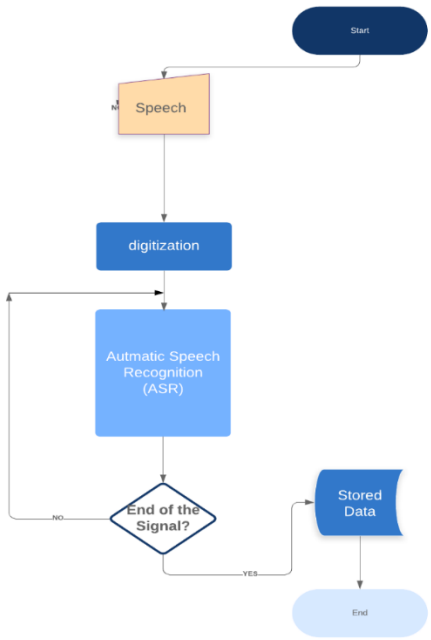


Fig. 5: Basic Operation of the Speech Recognition System

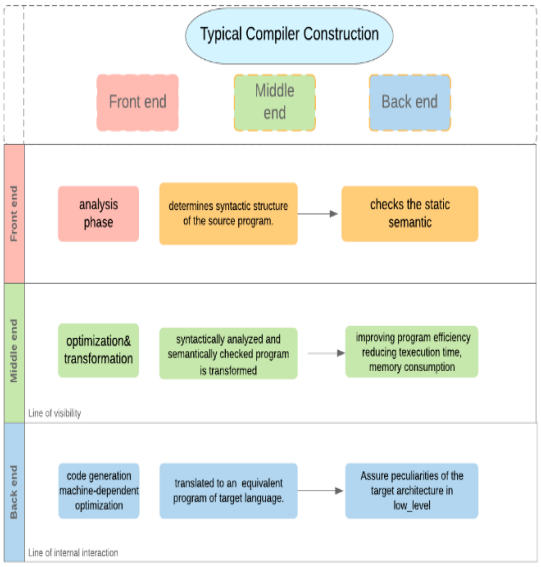


Fig. 6: Compiler Construction Operation

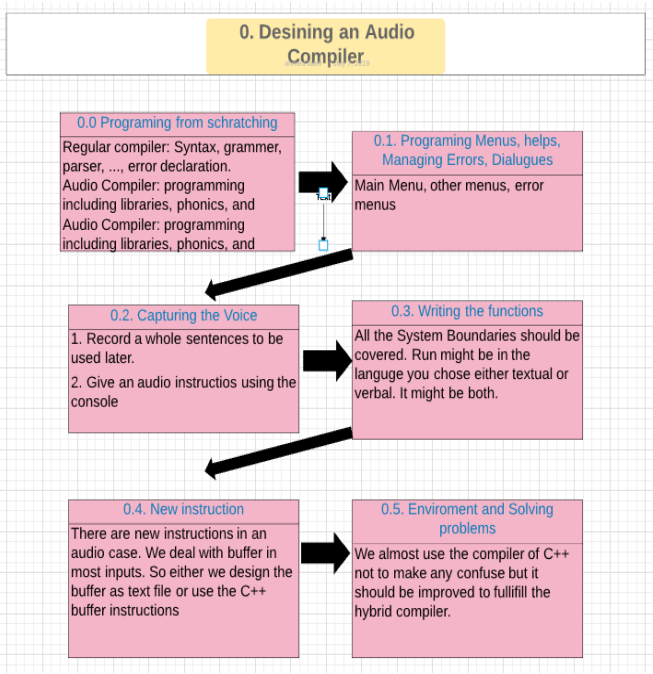


Fig. 7: Stages of an audio compiler

The design phase of this verbal compiler has been broken down into six stages called processors to outline the subject in general and the programming processes specifically, as seen in Fig. 8, shown above.

Stage 0.0. Programming from scratch

A programming language is a compiler or interpreter that contains many rules and procedures, libraries, and auxiliary operations to control the compiler. Yet, programmers can only identify problems and then program them. The innovation of our compiler in stage 0 is unlike programming languages in the past that received commands through keyboards, mice, and input tools in general. The resulting programming language enables the programmer to design and implement systems by voice commands. Stage 0 requires constructing a complete Compiler fully. Design a new compiler from scratch was one of the difficulties we faced. Implementing a compiler may take five years if only one programmer achieves it. We are not interested in developing a new programming language, but we need a modification to include sounds in the language for it to become more efficient.

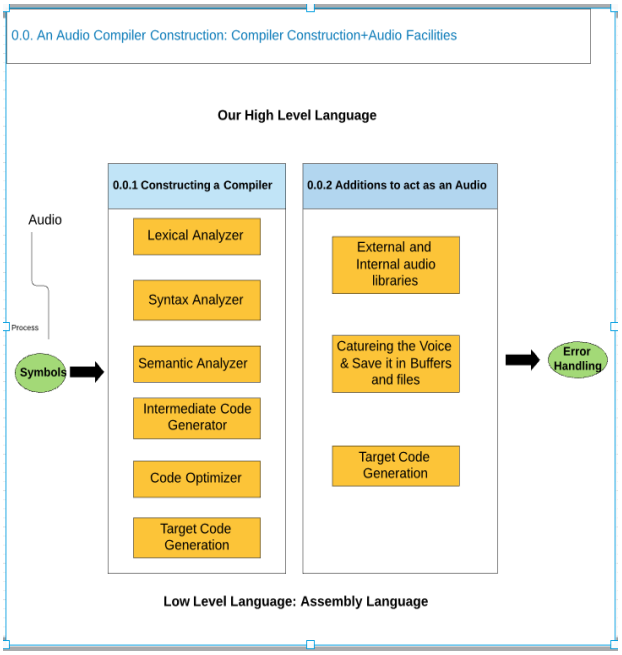
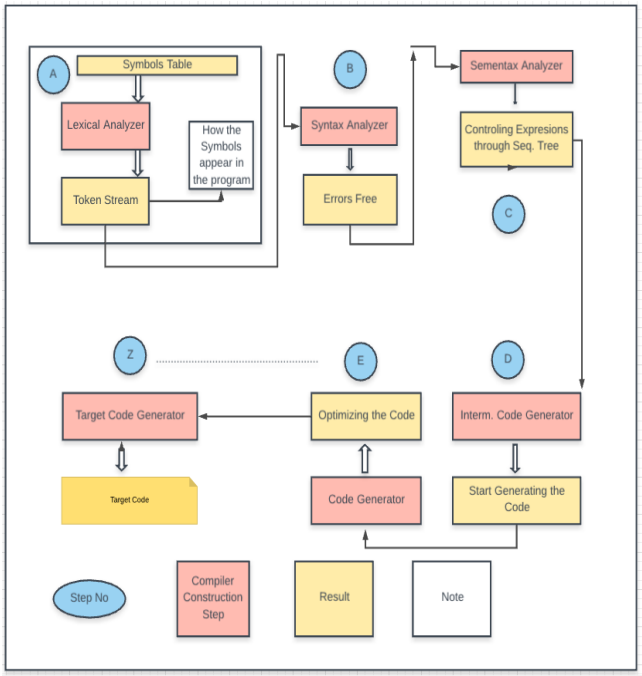


Fig. 9: For example, the part (processor) of an audio compiler could be broken down within two processors

The dilemma is that sound-related programming mostly depends on capturing sounds to save it temporarily in the Buffers. Two major problems must be confronted: A severe shortage of books that explain such types of programs and the limitation of the material on the Internet. Luckily, some fragmented material is provided by YouTubers who are interested in some applications.Buffers are audio streams stored for a while. A variable type of flowing data in buffer cannot be specified. Programmers must find a way to convert the buffers' contents into variables of the String type. This method only enables programmers to compare the variables that are defined in the programs with data in the Buffers. Programmers always have alternative solutions such as using text files. A Compiler acts as a calculator is designed: The desired process is verbally commanded be addition, which stored in the text file A as a variable of a string or an integer. The numbers that summed up through a regular program store in file B. Comparing the entered data in the text file "A" with the expected cases and matching with one of them is a required operation, and here is example of an "addition" of two numbers. Different operations can be programmed, but in this case, the issue is the sophistication and need for more text files. The previous explanation is a solution but is stressful for programmers and takes time to process.



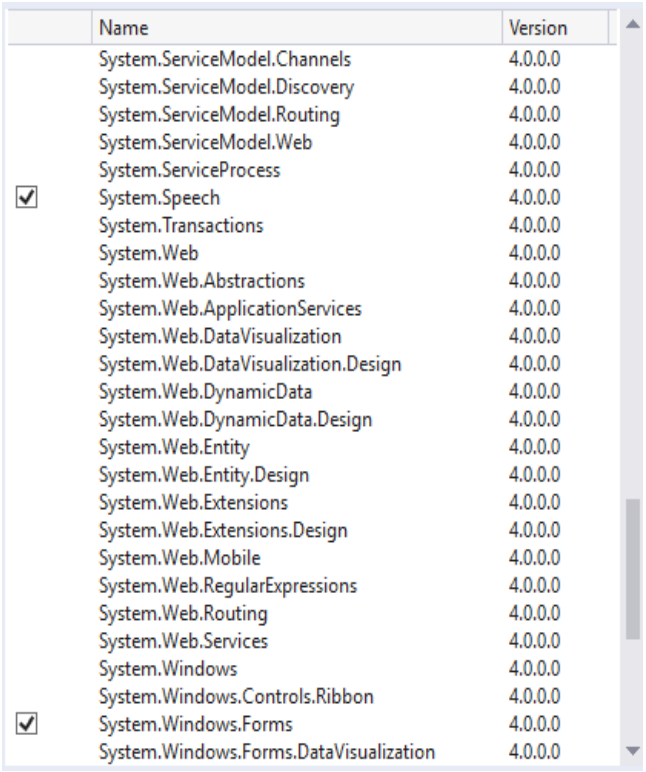


Fig. 10: More explanation about regular compiler's steps

III. Internal and External Libraries

One of the most important features and advantages offered by the programming languages is the various libraries, which are located within the language or programmed by the compiler, and they are called internal libraries. At the same time, there are additional libraries which serve specific purposes that are not available in the programming language, such libraries do not benefit all programmers in general. These libraries are called external libraries, and small software companies program them. External libraries are downloaded from professional sites. Several steps are followed to install them. Internal libraries are not commonly available in languages but are added from within the language interface. The language mainly has the necessary libraries to run in general, but the performance of some additional operations, such as the use of sound commands, needs to be added.

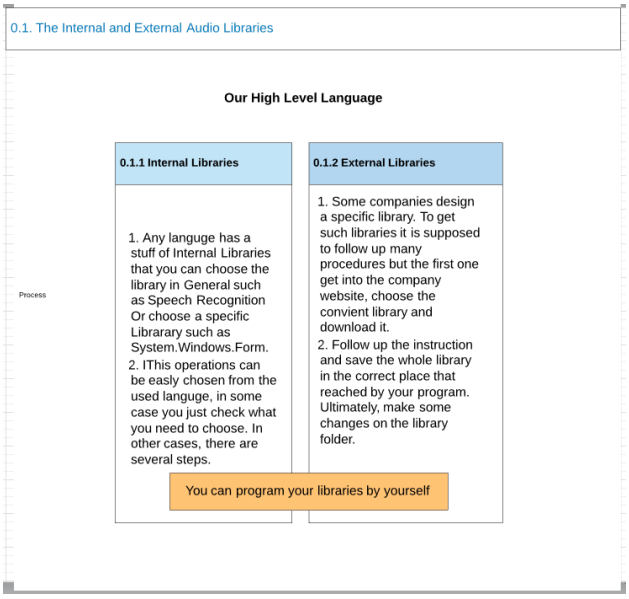


Fig. 11: The internal and external libraries.

Fig.12: An internal library

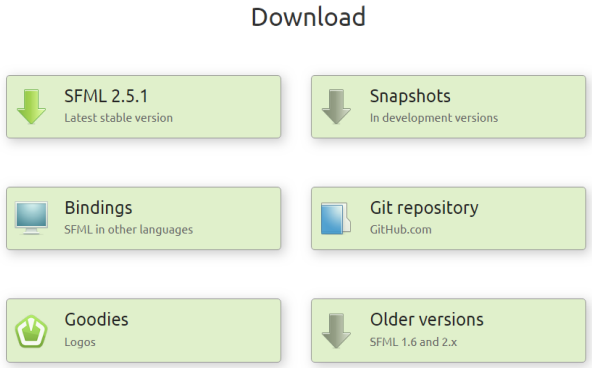


Fig. 13. a: External Library

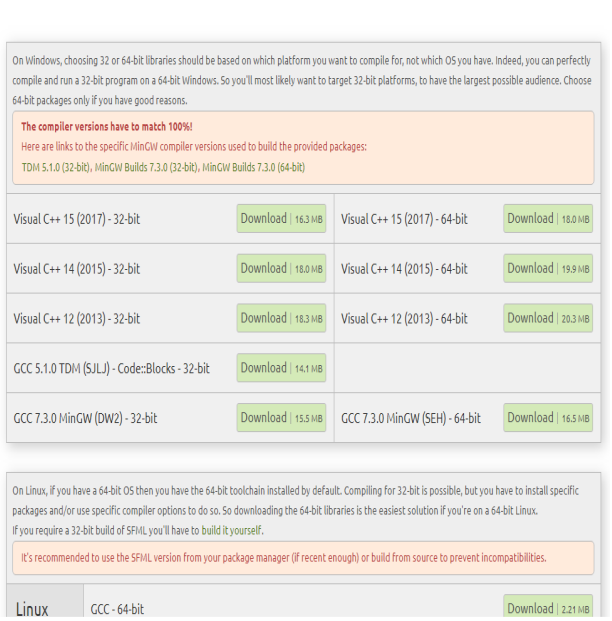


Fig. 13. b:

Fig. 11

Fig. 9, 10, and 11 clarifies how to download external library.

0.1.3. Error handling

In every language, when errors occur in one of the stages of compiling, the number, type, and line of the errors are shown. In this modified language, errors are classified and given numbers and displayed in interactive screens with programmers.

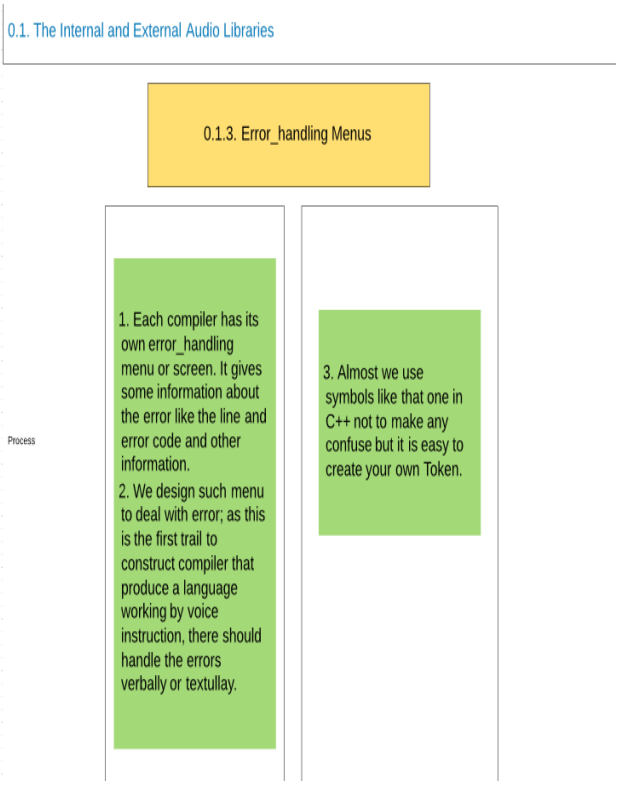


Fig. 14: Error handling Menus

In this case study, we analyze seven inputs and outputs of the six stages of translation for regular compiler, that begin from the input and are translated by a compiler stage. As shown in fig. 14 a. the statement is placed on a symbol table. Once the voice is captured, it is dealt with exactly as in the compiler stage. So, the input will become an audio. In fig. 14. b., the voice passes on the Syntax Analyzer that composes the tree if the capturing operation of the voice is correct; otherwise, an error occurs.

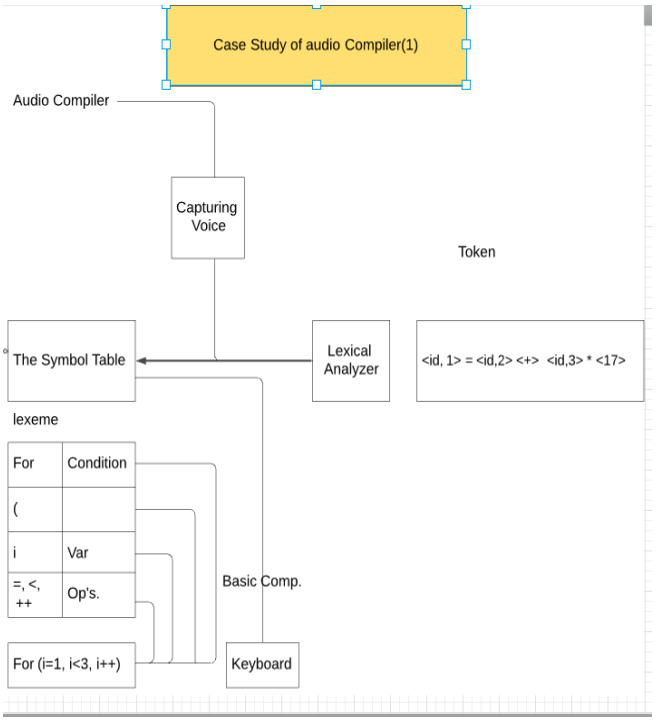


Fig.15. a

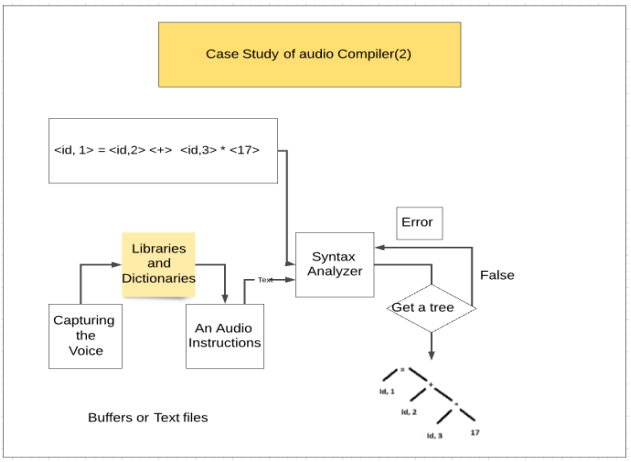


Fig. 15.b

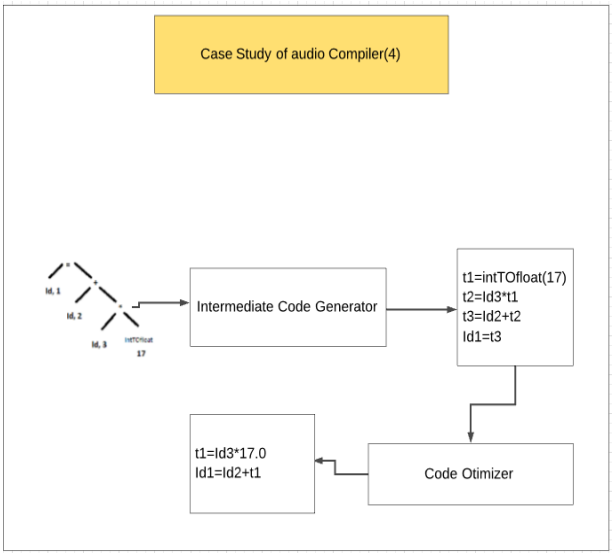


Fig. 16. a

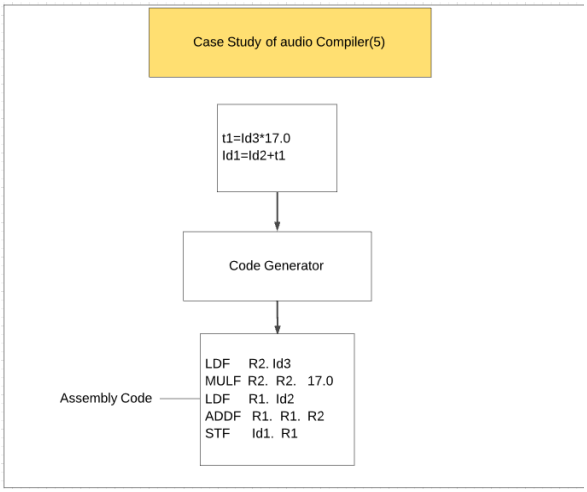


Fig. 16. b

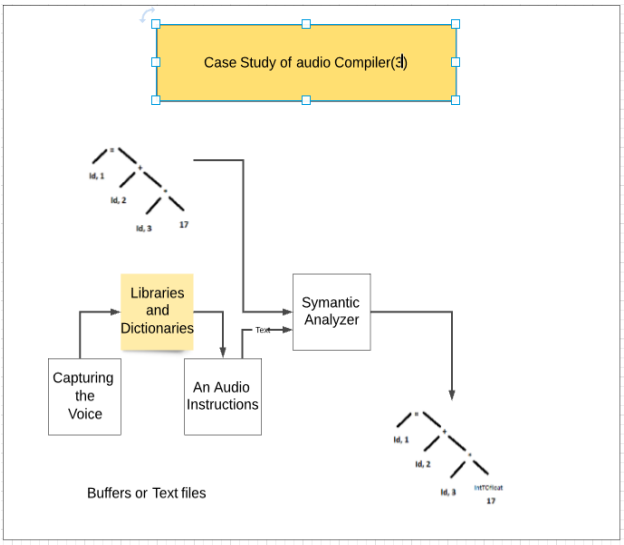


Fig. 16. c

Fig. 15, 16 clarifiy all the compiler construction stages. Fig 16 a, b, and c show the other four steps to reach to Machine Code (Assembly Code), ultimately. This code which deals with the contents of Microprocessors, Registers, and RAM and has access to other external resources such as memory. The example model of the compiler that works as a calculator receives audio commands and then works as a regular compiler, information that will be explained in detail later.

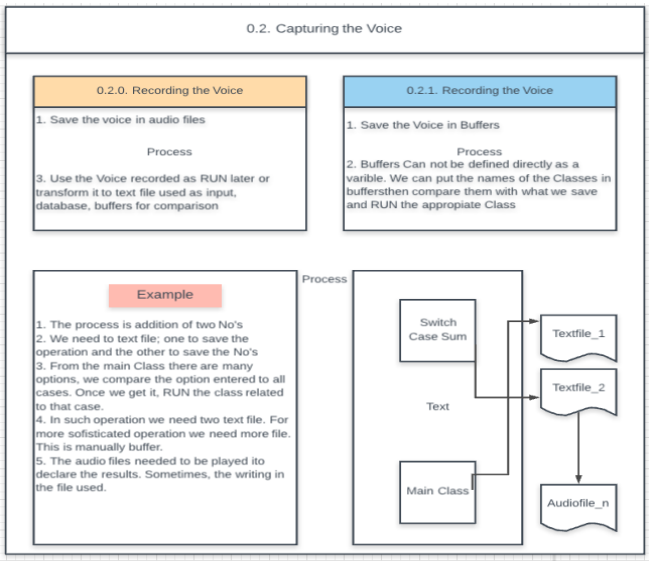


Fig. 17: An example of how to capture a voice and use it as a command

The compiler model that works as a calculator by receiving the instruction verbally.

In order not to confuse the users in a new language, all C ++ symbols have been used. For model programming, several options are taken into consideration. Despite the difficulty of C++ language and the lack of examples of all commands dealing with the sound, it is chosen. The most crucial advantage of C ++ is that it creates a code representing assembly language and thus has the required speed in dealing with the CPU. The most vital operating systems are programmed in C ++. In Fig. 15, the graph is considered Stage Three, the third processor of the whole system, designed to implement its operation into two subdivision Stages (Processors). This Phase is the first Stage after applying all the stages of the regular compiler. Audio programming allows many ways to capture the voice — categories which placed under two columns: First save the captured sound into any audio file format to deal with later or reuse the audio saved for other purposes. Second, save the voice in a temporary Buffers for a while. Both methods are interchangeably used as needed.

IV. Saving the Voice in an Audio File

An external library called SFML can be downloaded, and several procedures become executed. For instance, we downloaded some libraries from SMFL and stored them in the same folder where the language libraries are stored. The program will not work on another machine unless the same folders are stored in the same places and the same libraries have been transferred to these folders. Compile the file as an executed file to ensure that the program works correctly. Then the executed program is run in the objected file. Programmers are in the process of programming and moving between different computers encounter complications that occur because of the non-presence of external libraries in their folders. When we finish the previous procedure, using voice commands can record any audio file at any length. Audio files are useful in that they can record clarifications and introductions to show results and help menus. Furthermore, they declare system errors. One or two words can be recorded, transcribed, and stored in a text file to be used in the system. These text files can act as a database of the system while the column represents the "field" and the line represent the "record". The programmer can package classes to read data, and when the program runs, it deals with the DBs or text files. For example, if the "Addition" process is used, users command verbally to choose the "Add" procedure; the order is interpreted to be a variable such as an integer, a bit, or a string, and stored in an appropriate file. Based on the previous procedure, other functions is programmed that can claim variables to be saved in other text files, and recording audio files and playing them later in the proper program or typing the audio text inside the program to read can also be achieved. Adding two numbers, users give the command that reflects the summation and declare how many numbers to be added. Two files, in this case, are used, and if the operation is complicated, which means using more than a process like "Add" and "Square root", more than two files are used. These operations are expressed within the codes that classify them and direct each process to their procedure. The verbal interaction between users and machines continues until the calculation is finished, and the result s given in the form of reading the text stored or written, and then the number shown is read. The machine alerts users that the process is completed. Users must obtain the results in style written especially in major systems such as employee salary calculations. The input is done in the verbal form and under the control of the user. If the input is ambiguous, the use repeats to write what the command wants clearly. Ultimately, the input stored in the database or text files. However, when calculating the results, they should be shown in reports and the pay checks.

V. Using the Buffers' Technique

Inside such libraries, whether internal or external, there are instructions for buffering. The sound can be captured to the buffer as a stream and then stored as any variable. Using buffers is much easier, faster, and vital. One issue that requires attention is that the sound stored in the buffer does not become defined as a variable. Other commands address this problem by converting the sound into a String type variable. The sound in this format is very easily handled. The technique, identical to the steps discussed in the previous stage, has greater flexibility, faster speed, and no exit from the CPU to connect to files stored in other locations.

VI. Ambiguity

Whatever the language spoken, recognizing the speech does not only depend on listening to individual words that are taken from the context, but also involves watching the movement of the lips and body language. The speaker uses his knowledge and logical repository to associate sentences and distinguish words individually or in a related context. Moreover, even if unable to understand a vocabulary used in a sentence can expect it and know the meaning. The understanding is when the full context of the sentence is taken. Furthermore, the use of compound words, dialects, and idioms should be taken into consideration. When a computer or mobile phone is used to Recognize the Speech (RS), none of those as mentioned earlier, knowledge can help the devices. because the intelligence of computers is zero but many people think that a computer is a smart machine. Programmers must define everything each step in a logical form acceptable to the computer as a whole and programming languages as a part can be followed. As illustrated in our model, which is a verbal calculator, steps through which an integrated work can be created to deal with such an acoustic signal. Besides, noise and many different factors affect the quality of speech recognition techniques. Some of them are related to the computer itself, like the sounds it makes. External factors in the room where we work are included, such as the presence of backgrounds sounds or noise.

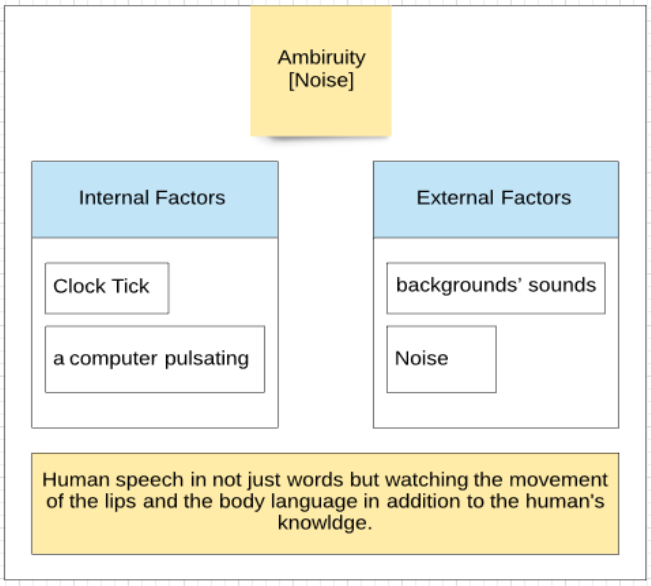


Fig. 18: The Ambiguity problem

VII. How to Solve Ambiguity

Many solutions can help users to specify a desired word. Programmers design a window in the program that gives matching options to the word. For example, if the word *two* is wanted, sometimes words *2*, *too*, and *tool* appears but users can correct the ambiguity manually by choosing the word that is needed. The same thing with the word *four*, which can be written as *4*, *for*, *ford*, *forth*, or *four* but this operation leads to little issues arise delay. Another solution that gives more accurate results but is difficult to be applied. Always the English language dictionary depends on two pronunciations for each word or at most four. A lady's and a man's voice with slow and fast speech can be heard. Even though speakers record different pronunciations for the same word and composing a whole new dictionary, is too hard. Instead, symbols that we use in the language in addition to the accompanying dictionary to the language ca e employed. Also, it can be facilitated by some kinds of dialogue like when the dictionary asks the user if a word is correct. If not, the wrong word is excluded while the user repeats pronouncing the word.

VIII. Conclusion

The previous steps indicate the importance of moving to programming languages that receive voice commands, and this leads to the design of a particular compiler to perform such tasks. Once a compiler is designed that performs limited functionality, then we can develop it to perform more comprehensive tasks. Artificial intelligence (AI) is beneficial in such systems. AR robotics programming is similar to programming like the calculator model. Another addition is hardware control should be improved. Our biggest problem is Ambiguity, which means that our pronunciation of the word does not seem sure to give the same spelling: The example of *two* or *four* that is mentioned before. In such case words like *to*, *too*, *2*, or any other similar spelling could be obtained. We find some solutions to the Ambiguity problem were found in many dictionaries with different speakers will help solve such a problem, and the dictionaries should be existing on software that handles audio-to-text SR.

Appreciation

Thanks to Miss Elyaa Issa for her drawing of Fig 1, 2, and 4.

References Références Referencias

1. Kepuska. Veton Z. "Speech and Language Processing Notes," Retrieved by 02/27/19 at 6:17 am from [http://my.fit.edu/~vkepuska/web/courses.php#ece5527- files]
2. Kepuska, Vetron website, retrieved on 02/28/19 @ 17:37 my.fit.edu/~vkepuska/web/
3. VK Kepuska, H.K. Ready. "Dynamic Time Wrapping Using Frequency Distributed Distances Measurements, "US Patent 6, 983, 246, 2006.
4. Kepuska, Veton Z., and Pattarapong Rojanasthien. "Speech corpus generation from DVDs of movies and TV serious." Journal of International Technology and Information Management, Vol. 20, no. 1, 2011.
5. Junqua, Jean-Claude, and Haton, Jean-Paul. "Robustness in Automatic Speech Recognition:" Fundamentals and Applications, Springer, Boston, MA, 1996 retrieved 03/28/2019 @ 9.11 am from [https://linkspringercom.portal.lib.fit.edu/content/pdf/10.1007%2F978-1-4613-1297-0.pdf].
6. Reinhard Wilhelm, and Helmut Seidl. "Sebastian Hack Compiler Design Syntactic and Semantic Analysis," Springer-Verlag Berlin Heidelberg 2013. Retrieved 03/28/2019 @ 12:12 pm from [https://link-springercom.portal.lib.fit.edu/content/pdf/10.1007%2F978-3-642-17540-4.pdf].
7. Rodriguez-Cartagena, Jean, Claudio-Palacios, Andrea K., Pacheco-Tallaj, Natalia, Santiago González, Valerie, Ordonez-Franco, Patricia. "The Implementation of a Vocabulary and Grammar for an Open-Source Speech Recognition Programming Platform," ASSETS '15 Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility, PP 447-448.
8. Richard Boulanger, Victor Lazzarini, et al. "The Audio Programming Book (The MIT Press)," Massachusetts Institute of Technology, 2011.
9. Allen I Holub. Compiler design in C.
10. Claudio Becchetti, Klucio Ricotti. "Speech Recognition: Theory and C++ Implementation"
11. Alfred Aho, Ravi Seth, and Jeffery D. Ullman. Compilers: Principles, Techniques, and tools"

12. Antony J. Dos Reis. "Compiler Construction: Using Java, JavaCC, and YaCC", Welly.
13. Charles N. Fischer, Richard J. LeBlank, Jr. "Creating A compiler WITH C"
14. Ronald Mac. "Writing Compilers and Intepreters: A Softeware Engineering Approach."
15. Mark Allen Wells. "Algorithms, Data structures, and Problem Solving with C++."
16. Will Pirkle. "*Designing Audio Effect Plug-Ins in C++: With Digital Audio Signal Processing Theory" pdf version, Focal Press, Taylor & Francis Groups, 2013*.
17. Richard Boulanger, Victor Lazzarini "The Audio Programming Book (The MIT Press)", Massachutess Institute of Technology, 2011.
18. Will Pirkle. "Designing Audio Effect Plugins in C++2nd Edition", Routledge Taylor and Friends Group, 2019.
19. Veton Z. Kepuska and Hussein A. Elharrati, "Robust Speech recognition System Using Conventional and Hybrid Features of MFCC, LPCC, PLP, RASTA-PLP and Hidden Markove Model Classifier in Noisy Conditions."
20. R.T. Salaja, R. Flynn, M. Russell. "Automatic *speech recognition* using artificial life," 25th IET Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communities Technologies (ISSC 2014/CIICT 2014), 2014, p. 91 – 95
21. M.J. Russell, K.M. Ponting, M.J. Tomlinson. "Measure of local speaking-rate for automatic *speech recognition*," IET, 1999.
22. D.S. Malik. "C++ Programming from Problem Analysis to Program Design", Eighth Edition
23. Kenneth C. Louden. "Compiler Construction: Principles and Practice 1st Edition."
24. Iyad Rahwan. "Argumentation in Artificial Intelligence. Springer, 2009.
25. Stuart Russell. "Human Compatible, ARTIFICIAL INTELLEGENCE AND THE PROBLEM OF CONTROL," KINDLE EDITION, VIKING, 2019.